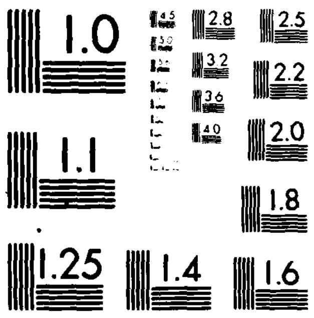
AD-A108 089 GEORGIA INST OF TECH ATLANTA PRODUCTION AND DISTRIBU--ETC F/G 12/1
ON THE COMPUTATIONAL COMPLEXITY OF STOCHASTIC SCHEDULING PROBLE--ETC(U)
SEP 81 M PINEDO N00014-80-K-0709
UNCLASSIFIED PDRC-81-11 NL

END
DATE
FILMED
1-82
DTIC


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL II

Georgia Institute of Technology

AD A108089

DTIC ELECTE DEC 3 1981 D A

PRODUCTION and DISTRIBUTION RESEARCH CENTER

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA, GEORGIA 30332

81 11 10 008

PDRC Report Series
No. 81-11
September 1981

ON THE COMPUTATIONAL COMPLEXITY OF STOCHASTIC SCHEDULING PROBLEMS

BY

Michael Pinedo[†]

Contract N00014-80-K-0709
NR-047-221

PDRC 81-11

† Georgia Institute of Technology
School of Industrial and Systems Engineering
Atlanta, Georgia 30332

This work was supported in part by the Office of Naval Research.

ON THE COMPUTATIONAL COMPLEXITY OF STOCHASTIC SCHEDULING PROBLEMS

Michael Pinedo

Georgia Institute of Technology

ABSTRACT

In this paper we consider stochastic scheduling models where all relevant data (like processing times, release dates, due dates, etc.) are independent random variables, exponentially distributed. We are interested in the computational complexity of determining optimal policies for these stochastic scheduling models. We give a number of examples of models in which the optimal policies can be determined by polynomial time algorithms while the deterministic counterparts of these models are NP-complete. We also give some examples of stochastic scheduling models for which there exists no polynomial time algorithm if P $\neq$ NP.

1. INTRODUCTION

In the last decade deterministic scheduling problems have received substantial attention. Two directions have always been very important in the investigation of a deterministic scheduling problem, namely:
(i) The search for algorithms that determine the optimal sequence efficiently, preferably in polynomial time.
(ii) The investigation of the computational complexity, i.e. determining whether or not the problem is NP-complete.
This research has yielded many results. An excellent survey can be found in Graham, Lawler, Lenstra, Rinnooy Kan (3). Noting that the deterministic distribution can be considered as a special case of an arbitrary stochastic distribution, many of these results can provide a key to the structure of the more general stochastic scheduling problems. Another special distribution is the *exponential* distribution. Stochastic scheduling models with exponentially distributed data (like processing times, release dates, due dates,

etc.) usually have a very nice structure, too.

In this paper whenever we refer to a specific stochastic scheduling problem we assume, unless otherwise specified, a problem with all data independent *exponentially* distributed. Of these scheduling models the objective functions usually have to be minimized in expectation. Our goal is either to obtain good (i.e. polynomial time) algorithms for determining the optimal policies within specific classes of policies or to show that no fast algorithms exist (assuming $P \neq NP$). We believe that it may be of interest to compare the computational complexity of these stochastic scheduling problems with the computational complexity of their deterministic counterparts. For a number of models the algorithms that determime the optimal policies in the deterministic and the stochastic versions are equally good. Examples of such models are $F2||E(C_{max})$ and $J2||(C_{max})$ where in the deterministic as well as in the stochastic version the algorithms are $O(n \log n)$. For some models there is a better algorithm for the deterministic version; an example is $O2||E(C_{max})$ where the algorithm for the deterministic version is $O(n)$, while the algorithm for the stochastic version is not as fast (see (11)). In Section 3 we give some examples of models of where the deterministic version is NP-complete and the stochastic version is very easy.

This paper is organized as follows: In Section 2 we give a short description of various classes of policies, formulations of stochastic scheduling problems as Markov Decision Processes, levels of complexity and how to determine the complexity of these stochastic scheduling problems. In Section 3 we present six models, exhibiting NP-completeness in the deterministic version, but having polynomial time algorithms in the stochastic setting. In Section 4 we present two examples of stochastic models for which we can show that there exists no polynomial time algorithm to determine the optimal policies (assuming $P \neq NP$). In Section 5 we discuss in which direction we intend to continue this research.

2. COMPLEXITY OF STOCHASTIC SCHEDULING PROBLEMS

In this section we first discuss three classes of policies. Afterwards we explain how we intend to investigate the computational complexity of determining optimal policies in any of these classes.

2.1. Classes of Policies

The first class of policies is the class of *static* policies. Here the decision-maker decides at $t = 0$ what his policy will be during the process; he is not allowed to make any subsequent changes. The optimal policy in this class often (but not always) prescribes a "list" schedule, i.e. whenever a machine becomes idle, the decision-maker starts processing the next job on the list independent of the state of the system at that moment. Consider the following example: $1|p_j \sim \exp(1), d_j \sim \exp(\mu_j)|E(\Sigma\, w_j U_j)$. As the processing times of all the jobs are exponentially distributed with mean one, the expected penalty caused by a particular job not meeting its due date only depends on the number of jobs that are to be processed before this one. Determining the optimal policy in the class of static policies reduces easily to a (deterministic) assignment problem. By solving this assignment problem the decision-maker obtains a list in which to process the n jobs. Determining the optimal policy in the class of static policies usually is equivalent to a deterministic problem. Hence, in determining whether or not this can be done in polynomial time, techniques similar to those used for deterministic scheduling problems can be used. Examples in Section 4 illustrate this point. Determining the optimal static policy is important because this optimal policy may be related to the optimal policy in another class of policies.

The second class of policies is the class of *nonpreemptive dynamic* policies. These can be described as follows: Under such a policy the decision-maker is allowed to determine his action at specified decision moments (e.g. when a job has finished its processing on a machine) making use of any newly available information. However, he is not allowed to interrupt the processing of any job already started.

The third class of policies is the class of *preemptive dynamic* policies. These are similar to our second class of policies, but now the decision-maker *is* allowed to interrupt the processing of any job at any time. Experience seems to show that determining the optimal policy in the class of preemptive dynamic policies is usually easier than determining the optimal policy in the class of nonpreemptive dynamic policies.

2.2 Markov Decision Processes and Levels of Complexity

A stochastic scheduling problem with all data exponentially distributed and in which the decision-maker is allowed at every decision moment to take all available information into account can be described as a Markov Decision Process in continuous time with a finite state space, finite action space and one absorbing state. It is a well known fact that for these MDP's there exists an optimal *stationary* policy. This formulation as an MDP can be done for the preemptive as well as for the nonpreemptive case.

For these MDP's we will make a distinction between three levels of complexity. Firstly, the ideal situation would be to have an algorithm that determines the optimal actions for *all* states simultaneously in polynomial time. Secondly, for some problems there may be an algorithm that determimes the optimal action in any arbitrary state in polynomial time, but that cannot determine the optimal actions in *all* states in polynomial time (the number of states often grows exponentially with the size of the problem). However, one has to keep in mind now that during its realization the process, from beginning to end, usually visits only a limited number of states, a number that usually is polynomial in the size of the problem. So, if the decision-maker has an algorithm that determines the optimal action in an arbitrary state in polynomial time and the number of times he uses this algorithm is polynomial in the size of the problem, the total number of computations the decision-maker has to perform during the process is still polynomial in the size of the problem. Gifford (2) has developed an algorithm for a stochastic scheduling problem where this is the case. Thirdly, for some problems it may not even be possible to develop an algorithm that determines the optimal action in an arbitrary state in polynomial time. The examples of Section 4 are of this kind.

2.3. Determining the Computational Complexity

We present here two approaches that may be useful when trying to determine the computational complexity of stochastic scheduling problems.

(i) *Formulation of the Markov Decision Process as a Linear Program.* It is a well known fact that determining the optimal actions in an MDP can be done via Linear Programming. It has been shown recently that there exists a polynomial time algorithm to solve an LP (5). This, of course, does not imply that there exists a polynomial time algorithm to determine the optimal actions in all states of an MDP, as the transformation from the MDP into an LP may not be possible in polynomial time. For most scheduling problems the size of the LP (the number of variables and constraints) grows exponentially in the size of the problem. However, this does *not* imply that there does not exist a polynomial time algorithm for the scheduling problem. So this approach appears to be useful only when the size of the LP grows polynomially in the size of the scheduling problem, because then we know we have a polynomial time algorithm. The following example illustrates this: Consider m machines and n jobs with precedence constraints. These precedence constraints have the form of $c(c > m)$ chains. The jobs have arbitrary independent exponential distributions, not necessarily identical. This MDP can be formulated as an LP for which it can be shown that, when c is fixed, the number of variables and constraints increase polynomially in n. So there exists an algorithm for determining the optimal actions that is polynomial in n. For the deterministic counterpart of this problem one can find easily an algorithm, based on dynamic programming, that is polynomial in n.

(ii) *Relating Optimal Dynamic Policies with Optimal Static Policies.* Determining whether the optimal static policy can be found in polynomial time is relatively easy, as this is basically a deterministic problem that can be approached with conventional techniques. Determining whether the optimal actions of an MDP can be found in polynomial time tends to be harder. However, it is possible to find special circumstances in which the optimal actions in a subset of the states have a specific one-to-one correspondence with the optimal list in the static problem. In case the optimal static policy cannot be determined in polynomial time we know that, if we indeed can establish a certain relationship between the actions of an optimal dynamic policy and the list given by the optimal static policy, the optimal dynamic policy cannot be determined in polynomial time either. The two examples of Section 4 illustrate this.

3. EXAMPLES OF STOCHASTIC SCHEDULING PROBLEMS THAT ARE EASY

In this section we discuss six models; the deterministic versions of five of these models are known to be NP-complete. Of the remaining one the complexity has not yet been determined, but the problem appears to be not too easy. The stochastic versions, with processing times exponentially distributed, of all of these models turn out to be very easy to analyze.

3.1. $P||C_{max}$ and its Stochastic Counterpart

In this model n jobs have to be scheduled on m identical parallel machines in order to minimize the makespan. Karp (4) showed that the deterministic version of this problem is NP-complete. Several researchers showed independently that when the processing times are exponentially distributed the Longest Expected Processing Time first (LEPT) policy minimizes the expected makespan. This LEPT policy is optimal in all three classes of policies discussed in Section 2.

3.2. $P2|res\ 1|C_{max}$ and its Stochastic Counterpart

In this model we have two machines and n jobs. There is one resource of which there is an amount s. Job j needs when being processed an amount s_j of this resource. At any time during the process the total amount of resource needed by the two jobs being processed, may not be larger than s. It is clear that there is no polynomial time algorithm for the deterministic version of this problem, as there exists no polynomial time algorithm for $P2||C_{max}$. Pinedo (13) considered the problem with exponentially distributed processing times. He assumed the following agreeability condition: Whenever for any two jobs $1/\lambda_i > 1/\lambda_j$ then $s_i \geq s_j$. A nonpreemptive policy was shown to be optimal in all three classes of policies. This nonpreemptive policy can be determined in O(n log n) time.

3.3. $1|d_j = d|\Sigma w_j U_j$ and its Stochastic Counterparts

In the deterministic version of this model we have n jobs, all with the same deadline. If job j does not meet this deadline the decision-maker incurs a penalty w_j. This problem is equivalent to the knapsack problem and therefore NP-complete (see Karp (4)). Of the stochastic version there are two variants: Firstly, all deadlines are i.i.d. random variables with arbitrary distribution (not necessarily exponential). Secondly, all jobs have the same deadline and this deadline is an arbitrarily distributed random variable.

Derman et al (1) have shown that in order to minimize the expected objective in all three classes of policies for the second variant the decision-maker has to order the jobs in decreasing order of $w_j \lambda_j$ where $1/\lambda_j$ is the mean of the exponentially distributed processing time of job j. Pinedo (12) showed that for the first variant the same policy minimizes the expected objective in the class of static policies.

3.4. $P|d_j = d|\Sigma U_j$ and its Stochastic Counterparts

This model is similar to the model discussed in 3.3. The only difference lies in the fact that instead of one machine we now have m machines in parallel. It is clear that there exists no polynomial time algorithm for the deterministic version of this problem either. Of the stochastic version with exponentially distributed processing times there are again two variants: Firstly all deadlines are i.i.d. random variables with arbitrary distribution (not necessarily exponential). Secondly, all jobs have the same deadline and this deadline is an arbitrarily distributed random variable. Pinedo (12) has shown that for the first variant the policy that schedules the task in increasing order of their expected processing times minimizes the expected objective in the class of static policies provided the distribution function of the deadline is concave. He also showed that for the second variant the same policy is optimal in all three classes of policies, again provided the distribution function being concave.

3.5. $1|d_j = d|\Sigma w_j T_j$ and its Stochastic Counterparts

In the deterministic version of this model all jobs have again the same deadline. If job j does not meet the deadline the decision-maker incurs a penalty $w_j T_j (= w_j(C_j - d_j))$. The computational complexity of this problem has not yet been determined, but Lawler and Moore (8) developed a pseudo-polynomial algorithm for this problem. Again two variants have been considered for the stochastic version. Firstly, all deadlines are i.i.d. random variables, with arbitrary distribution. Secondly, all jobs have the same deadline with arbitrary distribution. Pinedo (12) showed that for both variants the optimal policy in the class of static policies instructs the decision-maker to schedule the jobs in decreasing order of $w_j \lambda_j$. This policy is also optimal in the class of dynamic policies for the second variant. In (12) more general models are discussed; models where the distributions of the deadlines are not identical but "agreeable" in the following sense: whenever $w_j \lambda_j > w_i \lambda_i$ the distribution of the deadline of job j is stochastically smaller than the distribution of the deadline of job i.

3.6. $1|pmtn, r_j|\Sigma w_j C_j$ and its Stochastic Counterpart

In the deterministic version there are n jobs released at given release dates and the sum of the weighted completion times has to be minimized. Labetoulle et al (6) have shown that this problem is NP-complete. Pinedo (12) considered the stochastic version where the release dates have an arbitrary joint distribution. The class of preemptive dynamic policies was considered. In this class the following policy was shown to be optimal: At any point in time, of the jobs that already have been released the one with the highest value of $w_j \lambda_j$ is released, the machine has to be preempted and this new job has to be put on the machine.

3.7. Additional Remarks

(i) In the stochastic models of 3.3 and 3.5 it is possible to include release dates in such a way that the optimal preemptive dynamic policy still can be determined easily. Consider for example the following extension: All due dates are identical and the release dates may be any combination of time epochs prior to this due date.

(ii) Gifford (2) considered the models of 3.2 and 3.3 with due dates nonidentically exponentially distributed. He developed algorithms which in some cases are polynomial time, in other cases not.

(iii) There are of course more models of which the deterministic version is NP-complete and for which in the stochastic setting there exists a polynomial time algorithm. Pinedo (10) and Weiss (14) considered a problem in reliability theory where this is the case.

4. EXAMPLES OF STOCHASTIC SCHEDULING PROBLEMS THAT ARE HARD

In this section we present two examples of stochastic scheduling problems for which we can show that, if $P \neq NP$, no polynomial time algorithm exists to determine the optimal policies in any of the three classes of policies mentioned in Section 2. In our approach we follow the method described in Section 2, i.e. we first prove that the static problem cannot be solved in polynomial time, after that we show that there exists a one-to-one correspondence between the optimal list of the static problem and the optimal actions of an appropriately chosen subset of states in the preemptive or nonpreemptive dynamic problems. We will not give rigorous proofs in this section, as the technical details are rather lengthy; the complete proofs will appear elsewhere. In both examples precedence constraints play an important role; without these precedence constraints the problems are easy. The first example is a rather straightforward extension of a deterministic result. The second example is more complicated.

4.1 $\underline{1|prec,\ p_j \sim exp(1)|E(\Sigma\ w_jC_j)}$

In this model we have one machine and n jobs, each of these jobs having a processing time exponentially distributed with mean one. There are precedence constraints which may have an arbitrary form. The expected sum of the weighted completion times has to be minimized. Lawler (7) considered the case where all the jobs have identical deterministic processing times, i.e. $1|prec,\ p_j = 1|\ \Sigma\ w_jC_j$. He showed through a reduction from LINEAR ARRANGEMENT that this problem is NP-complete even if $w_j \varepsilon \{1,2,3\}$ for all j. One can show that changing the processing times of the jobs from deterministic with mean one to exponential with mean one does not change the optimal sequence of the jobs and that this sequence is an optimal policy in all three classes of policies. This implies that, if $P \neq NP$, there exists no polynomial time algorithm to determine the optimal policy for this stochastic scheduling problem.

4.2. $\underline{1|prec,\ p_j \sim exp(p),\ d_j \sim exp(\mu_j)|E(\Sigma\ U_j)}$.

We have again one machine and n jobs. Job j has an exponentially distributed processing time with mean $1/p$ and an exponentially distributed due date with mean $1/\mu_j$. There are precedence constraints and the expected number of jobs overdue has to be minimized. We first will show that finding the optimal static policy cannot be done in polynomial time. Suppose job j is the k^{th} job to be processed. The probability that job j will not meet its deadline is $1 - (p/(p+\mu_j))^k$. Now let $1/\mu_j >> n/p$ for all j. The probability that job j will not meet its deadline is then approximately $\mu_j \cdot k/p$. The objective to be minimized is the expected number of jobs that do not meet their deadline; this may be approximated by $\Sigma\ \mu_j \cdot E(C_j)$ where $E(C_j)$ is the expected completion time of job j. This objective, however, after replacing μ_j with w_j, is identical to Lawler's objective in the problem $1|prec, p_j \sim exp(p)|E(\Sigma\ U_j)$, when $1/\mu_j >> n/p$ for all j, results in a sequence that is identical to an optimal sequence in Lawler's $1|prec, p_j = 1|\Sigma\ w_j C_j$. This implies that the optimal static policy cannot be determined in polynomial time.

Before investigating the optimal dynamic policies for $1|prec, p_j \sim exp(p)|E(\Sigma\ U_j)$, we consider the following model, which at first sight may seem unrelated: We have one machine and n jobs with identical deterministic processing times of one time unit. Of only one of these jobs a due date will occur during $[0,n]$; of n-1 jobs no due date will occur. The due date will be job j's with probability q_j. This due date is uniformly distributed over $[0,n]$. There are arbitrary precedence constraints. The objective is to maximize the probability that no job will be overdue. It can be shown that the optimal static policy in this model is equivalent to an optimal sequence in Lawler's deterministic problem $1|prec, p_j = 1|\Sigma\ w_j C_j$ (the q_j's play the role of the w_j's). It can also be shown that this optimal static policy is also optimal in the classes of preemptive and nonpreemptive dynamic policies. In the following paragraph we will refer to the above model as the One Due Date Model.

Now we consider the dynamic versions of $1|prec, p_j \sim exp(p), d_j \sim exp(\mu_j)|E(\Sigma\ U_j)$. Let $\Sigma\ \mu_i = \alpha$, where α is small and $\alpha n/p$ is very small. Making the rates $\mu_i, i=1,\ldots,n$ small has the following two effects: Firstly, the event of any due date occurring during the time needed to process the n jobs has a very low probability. Secondly, if a due date does occur, the time it occurs is approximately uniformly distributed over $[0,T]$, where T is the random time needed to process all jobs. It can be shown that, for n sufficiently large, the probability of no due date occurring during $[0,T]$ is $1 - \alpha n/p + O((\alpha n/p)^2)$, the probability of one due date occurring is $\alpha n/p + O((\alpha n/p)^2)$ and the probability of two or more due dates occurring is $O((\alpha n/p)^2)$. The problem can be formulated as an MDP in continuous time; optimal actions have to be determined in all possible states, a state being characterized by the set of jobs already processed and the set of due dates already occurred. It can be shown through some rather technical bounding arguments that in order to determine the optimal action in a state one may base one's decision solely on the event of exactly one due date occurring in the remaining time needed to finish the process. This is intuitive as in the case of no due dates occurring any sequence would be optimal, and the probability of two or more due dates occurring is very small in comparison with the probability of one due date occurring. But the problem conditional on one due date occurring is approximately equivalent to the One Due Date Model. Suppose the optimal job sequence for the One Due Date Model is $j_1, j_2, \ldots, j_n$ (which represents simultaneously the optimal static policy as well as the optimal preemptive and nonpreemptive dynamic policy) then it can be shown, through bounding arguments, that when the MDP is in a state with jobs $j_1, j_2, \ldots, j_k$ finished and all due dates still to come, the optimal action is to start job j_{k+1}. Now one can reason that if there exists a polynomial time algorithm to determine the optimal actions in the states of the MDP there also exists a polynomial time algorithm for model M. As, if $P \neq NP$, there exists no polynomial time algorithm for model M, there does not exist a polynomial time algorithm for the MDP either.

5. CONCLUSION

This paper should provide an indication of the direction in which we are currently working. To this extent, it constitutes a sort of progress report. Presently, we are dealing with stochastic scheduling problems with due dates; this research is far from complete. However, Gifford (2), as suggested before, has obtained some interesting results with regard to algorithms for these problems. The following aspects of these due date problems are currently under investigation: The influence of processing time distributions other than exponential (we are thinking here mainly of distributions with a Decreasing Failure Rate (DFR)). In addition, the influence of special classes of precedence constraints, namely series-parallel (see (7)) or chains (see (8)) is also worthy of investigation as well as the development of bounds on the expected values of the objective functions when processing times are exponential.

ACKNOWLEDGMENT

I am grateful to Ted Gifford for his many helpful comments and Betty Plummer for her excellent typing.

REFERENCES

(1) Derman, C., Lieberman, G. and Ross, S., "A Renewal Decision Problem": 1978, Management Sci. 24, pp. 554-561.

(2) Gifford, T., "Algorithms for Stochastic Scheduling Problems with Due Dates": in preparation.

(3) Graham, R., Lawler, E.L., Lenstra, J.K, and Rinnooy Kan, A.H.G., "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey": 1979, Ann. Discrete Math. 5, pp. 287-326.

(4) Karp, R.M., "Reducibility Among Combinatorial Problems": 1972, R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York), pp. 85-103.
(5) Khachian, L.G., "A Polynomial Algorithm in Linear Programming": 1979, Doklady, Akademii Nauk SSSR 224, pp. 191-194.
(6) Labetoulle, T., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G., "Preemptive Scheduling of Uniform Machines Subject to Release Dates": 1979, Technical Report Mathematisch Centrum.
(7) Lawler, E.L., "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints": 1978, Ann. Discrete Math. 2, pp. 75-90.
(8) Lawler, E.L. and J.W. Moore, "A Functional Equation and its Application to Resource Allocation and Sequencing Problems": 1969, Management Science, 16, pp. 77-84.
(9) Lenstra, J.K. and Rinnooy Kan, A.H.G., "Complexity Results for Scheduling Chains on a Single Machine": 1980, Eur. J. of Operational Research 4, pp. 270-275.
(10) Pinedo, M.L., "Scheduling Spares with Exponential Lifetimes in a Two Component Parallel System": 1980, J. of Appl. Prob. 17, pp. 1025-1032.
(11) Pinedo, M.L. and Ross, S.M., "Minimizing the Expected Makespan in Stochastic Open Shops": 1980, in editorial process, J. of Appl. Prob.
(12) Pinedo, M.L., "Comparisons between Deterministic and Stochastic Scheduling Problems with Release Dates and Due Dates": 1981, in editorial process, Operations Research.
(13) Pinedo, M.L., "On Stochastic Scheduling with Resource Constraints": 1981, in editorial process, Management Science.
(14) Weiss, G., "Scheduling Spares with Exponential Lifetimes in a Two Component Parallel System": 1981, in editorial process, Nav. Res. Logistics Quart.

FILMED
8